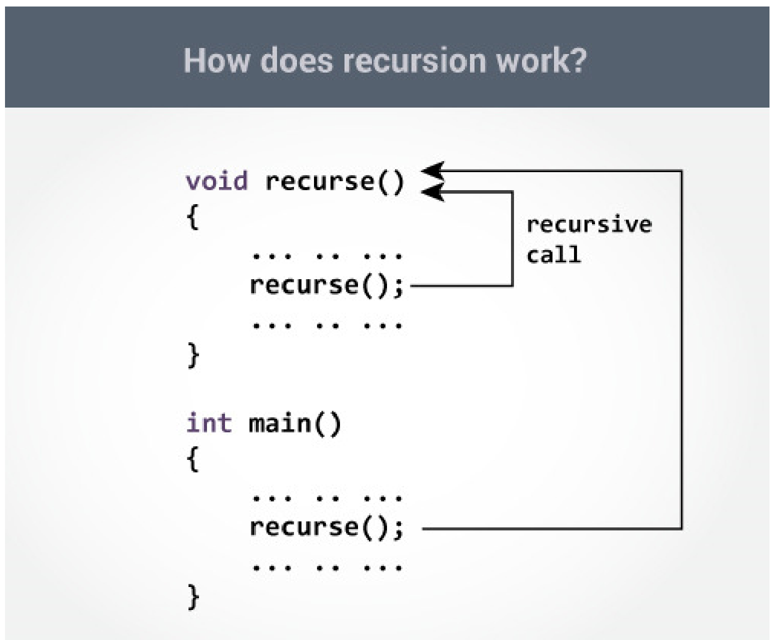
Structured Programming Language (CSE113)
Lecture #07
Recursion in C Programming

A function that calls itself is known as a **recursive function**. And, this technique is known as **recursion**.
Recursive function is very useful to solve many mathematical problems like to calculate factorial of a number, generating Fibonacci series, etc.



The recursion continues until some condition is met to prevent it.

To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and other doesn't.

Advantages and Disadvantages of Recursion

- **Advantages**: Recursion makes program elegant and cleaner. All algorithms can be defined recursively which makes it easier to visualize and prove.
- **Disadvantages**: If the speed of the program is vital then, you should avoid using recursion. Recursions use more memory and are generally slow. Instead, you can use loop.

Example 1: Factorial of a Number Using Recursion

```
#include<stdio.h>

int fact(int n);

int main()
{
    int n, k;

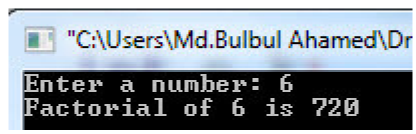
    printf("Enter the value of n : ");
    scanf("%d", &n);

        k = fact(n);

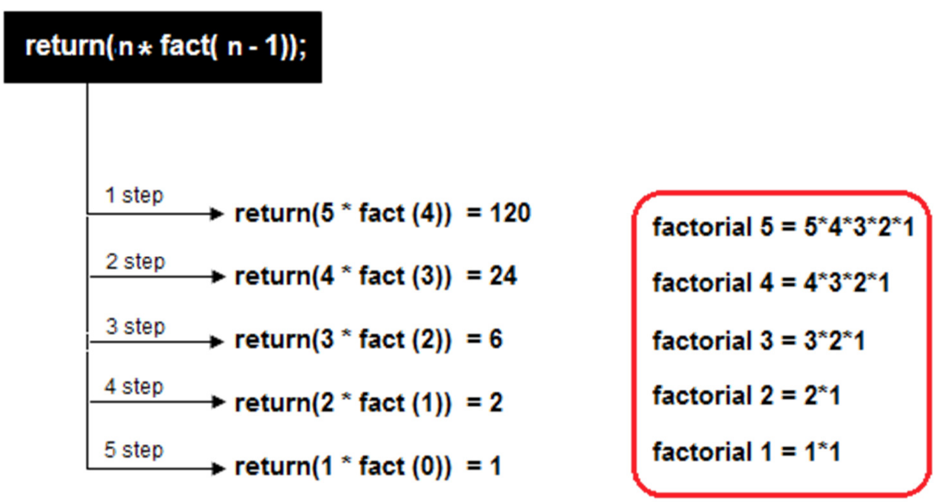
    printf("Factorial of  %d is %d\n", n,k);

    return 0;
}
```

```
int fact(int n){
    if(n == 0)
        return 1;
    else
        return n * fact(n-1);
}
```



Factorial of 5: Operational steps



Example 2: Find the Nth Fibonacci Number using Recursion

```
#include<stdio.h>

int fact(int n);

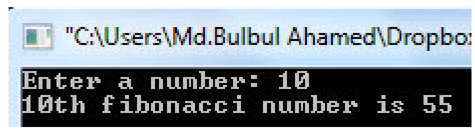
int main()
{
   int n, k;
   printf("Enter the value of n : ");
   scanf("%d", &n);

   k = fibo(n);

   printf("%dth Fibonacci number is %d\n",n,  k);

   return 0;
}
```

```
int fibo(int n){
   if(n == 1 || n == 2)
      return 1;
   else
      return fibo(n-1) + fibo(n-2);
}
```



Example 3: Print Fibonacci Series using Recursion

```
#include<stdio.h>

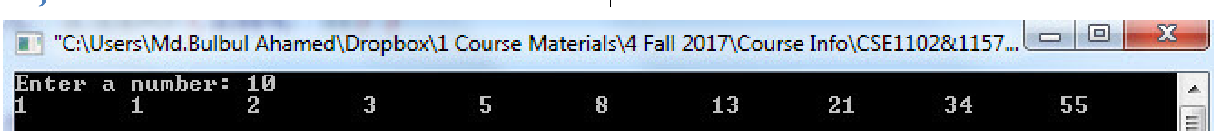
int fact(int n);

int main()
{
   int n,k,i;

   printf("Enter the value of n : ");
   scanf("%d",&n);

   for(i=1; i<= n; i++){
        k = fibo(i);
      printf("%d\t",k);
   }
   return 0;
}
```

```
int fibo(int n){
   if(n == 1 || n == 2)
      return 1;
   else
      return fibo(n-1) + fibo(n-2);
}
```



Example 4: Sum of Natural Numbers Using Recursion

```
#include<stdio.h>

int sum(int num);

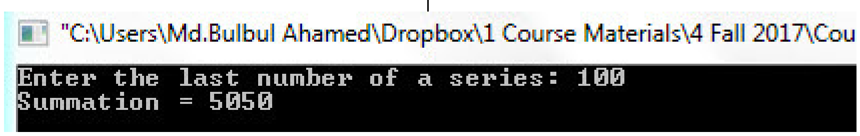
int main()
{
    int num, k;

    printf("Enter a number : ");
    scanf("%d", &num);

    k = sum(num);

    printf("%d", k);
    return 0;
}
```

```
int sum(int num){
    if(num == 0)
        return 0;
    else
        return num + sum(num-1);
}
```



Example 5: Display series (1, 2, 3, 4, 5 , … … , n)

```
#include<stdio.h>

int series(int n);

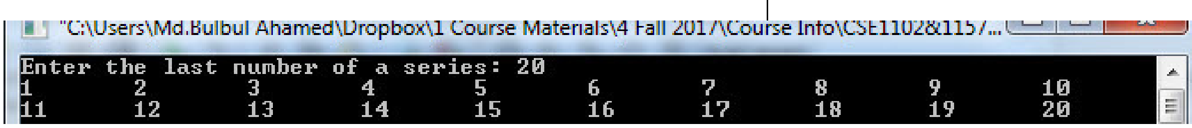
int main(){
    int n;

    printf("Enter the last number of a series: ");
    scanf("%d",&n);

    series(n);

    return 0;
}
```

```
int series(int n){
    int k;
    if(n == 0) return 0;
     series(n-1);
    printf("%d\t",n);
}
```



Example 6: Display series (n, n-1, … … , 5, 4, 3, 2, 1)

```
#include<stdio.h>

int series(int n);

int main(){
   int n;

   printf("Enter the last number of a series: ");
   scanf("%d",&n);

   series(n);

   return 0;
}
```

```
int series(int n){
   int k;
   if(n == 0) return 0;
   printf("%d\t",n);
     series(n-1);
}
```

```
"C:\Users\Md.Bulbul Ahamed\Dropbox\1 Course Materials\4 Fall 2017\Course Info\CSE1102&1157...
Enter the last number of a series: 20
20      19      18      17      16      15      14      13      12      11
10      9       8       7       6       5       4       3       2       1
```

Note: In order to prevent infinite recursive call, we need to define proper exit condition in a recursive function.

Example: C Program to show infinite recursive function

```
#include<stdio.h>

int main()
{
   printf("Hello world");
   main();
   return 0;
}
```

In this program, we are calling ***main()*** from ***main()*** which is recursion. But we haven't defined any condition for the program to exit. Hence this code will print "**Hello world**" infinitely in the output screen.